Express Mail Label No. EV396083740US

[0001] METHOD AND SYSTEM FOR RESTORING A VOLUME IN A CONTINUOUS DATA PROTECTION SYSTEM

[0002] CROSS REFERENCE TO RELATED APPLICATION(S)

[0003] This application claims priority from U.S. Provisional Application No. 60/---,---, entitled "METHOD AND SYSTEM FOR CONTINUOUS DATA PROTECTION," filed on February 4, 2004; and U.S. Provisional Application No. 60/---,---, entitled "CONTINUOUS DATA PROTECTION IN A COMPUTER SYSTEM," filed on February 5, 2004, which are incorporated by reference as if fully set forth herein.

[0004] FIELD OF INVENTION

[0005] The present invention relates generally to continuous data protection, and more particularly, to restoring a volume in a continuous data protection system.

[0006] BACKGROUND

[0007] Hardware redundancy schemes have traditionally been used in enterprise environments to protect against component failures. Redundant arrays of independent disks (RAID) have been implemented successfully to assure continued access to data even in the event of one or more media failures (depending on the RAID Level). Unfortunately, hardware redundancy schemes are ineffective in dealing with logical data loss or corruption. For example, an accidental file deletion or virus infection is automatically replicated to all of the redundant hardware components and can neither be prevented nor recovered from by such technologies. To overcome this problem, backup technologies have traditionally been deployed to retain multiple versions of a production system over time. This allowed administrators to restore previous versions of data and to recover from data corruption.

[0008] Backup copies are generally policy-based, are tied to a periodic schedule, and reflect the state of a primary volume (i.e., a protected volume) at the particular point in time that is captured. Because backups are not made on a continuous basis, there will be some data loss during the restoration, resulting from a gap between the time when the backup was performed and the restore point that is required. This gap can be significant in typical environments where backups are only performed once per day. In a mission-critical setting, such a data loss can be catastrophic. Beyond the potential data loss, restoring a primary volume from a backup system can be complicated and often takes many hours to complete. This additional downtime further exacerbates the problems associated with a logical data loss.

[0009] The traditional process of backing up data to tape media is time driven and time dependent. That is, a backup process typically is run at regular intervals and covers a certain period of time. For example, a full system backup may be run once a week on a weekend, and incremental backups may be run every weekday during an overnight backup window that starts after the close of business and ends before the next business day. These individual backups are then saved for a predetermined period of time, according to a retention policy. In order to conserve tape media and storage space, older backups are gradually faded out and replaced by newer backups. Further to the above example, after a full weekly backup is completed, the daily incremental backups for the preceding week may be discarded, and each weekly backup may be maintained for a few months, to be replaced by monthly backups. The daily backups are typically not all discarded on the same day. Instead, the Monday backup set is overwritten on Monday, the Tuesday backup set is overwritten on Tuesday, and so on. This ensures that a backup set is available that is within eight business hours of any corruption that may have occurred in the past week.

[0010] Despite frequent hardware failures and the necessity of ongoing maintenance and tuning, the backup creation process can be automated, while restoring data from a backup remains a manual and time-critical process. First, the appropriate backup tapes need to be located, including the latest full backup and any

incremental backups made since the last full backup. In the event that only a partial restoration is required, locating the appropriate backup tape can take just as long. Once the backup tapes are located, they must be restored to the primary volume. Even under the best of circumstances, this type of backup and restore process cannot guarantee high availability of data.

[0011] Another type of data protection involves making point in time (PIT) copies of data. A first type of PIT copy is a hardware-based PIT copy, which is a mirror of the primary volume onto a secondary volume. The main drawbacks to a hardware-based PIT copy are that the data ages quickly and that each copy takes up as much disk space as the primary volume. A software-based PIT, typically called a "snapshot," is a "picture" of a volume at the block level or a file system at the operating system level. Various types of software-based PITs exist, and most are tied to a particular platform, operating system, or file system. These snapshots also have drawbacks, including occupying additional space on the primary volume, rapid aging, and possible dependencies on data stored on the primary volume wherein data corruption on the primary volume leads to corruption of the snapshot. In addition, snapshot systems generally do not offer the flexibility in scheduling and expiring snapshots that backup software provides.

[0012] While both hardware-based and software-based PIT techniques reduce the dependency on the backup window, they still require the traditional tape-based backup and restore process to move data from disk to tape media and to manage the different versions of data. This dependency on legacy backup applications and processes is a significant drawback of these technologies. Furthermore, like traditional tape-based backup and restore processes, PIT copies are made at discrete moments in time, thereby limiting any restores that are performed to the points in time at which PIT copies have been made.

[0013] A need therefore exists for a system that combines the advantages of tape-based systems with the advantages of snapshot systems and eliminates the limitations described above.

[0014] SUMMARY

[0015] A method for synchronizing a secondary volume with a primary volume in a continuous data protection system begins by scanning a region of the primary volume. The scanned region is then compared with a corresponding region of the secondary volume. An identification of the scanned region is stored in a compare delta map when the comparison results in a discrepancy between the scanned region and the corresponding region. Data is copied from the primary volume to the secondary volume, using the compare delta map as a guide to locate the data to copy. If the data protection system has failed just prior to the restore process being initiated, it is termed a re-baseline, and the entire primary volume is scanned. If the data protection system is active prior to the restore process being initiated, it is termed a re-synchronization, and is optimized by scanning only select regions of the primary volume. A dirty region log is used to maintain a list of those regions of the primary volume that are to be scanned.

[0016] A method for restoring a primary volume from a secondary volume in a continuous data protection system includes the steps of selecting a snapshot of the primary volume to be restored and loading the snapshot from the secondary volume to the primary volume.

[0017] A system for synchronizing a secondary volume with a primary volume in a continuous data protection system includes scanning means for scanning a region of the primary volume; comparing means for comparing the scanned region with a corresponding region of the secondary volume; storing means for storing an identification of the scanned region in a compare delta map when the comparing means returns a discrepancy between the scanned region and the corresponding region; and copying means for copying data from the primary volume to the secondary volume, using the compare delta map as a guide to locate the data to copy. If the data protection system has failed just prior to the restore process being initiated, it is termed a re-baseline, and the entire primary volume is scanned. If the data protection

system is active prior to the restore process being initiated, it is termed a re-synchronization, and is optimized by scanning only select regions of the primary volume. A dirty region log is used to maintain a list of those regions of the primary volume that are to be scanned.

[0018] A system for restoring a primary volume from a secondary volume in a continuous data protection system includes at least one snapshot of the primary volume, each snapshot corresponding to a different point in time; selecting means for selecting a snapshot to be restored; and loading means for loading the selected snapshot from the secondary volume to the primary volume.

[0019] BRIEF DESCRIPTION OF THE DRAWINGS

[0020] A more detailed understanding of the invention may be had from the following description of a preferred embodiment, given by way of example, and to be understood in conjunction with the accompanying drawings, wherein:

[0021] Figures 1A-1C are block diagrams showing a continuous data protection environment in accordance with the present invention;

[0022] Figure 2 is an example of a delta map in accordance with the present invention;

[0023] Figures 3A-3C are flowcharts of a re-baseline procedure in accordance with the present invention; and

[0024] Figure 4 is a diagram of a delta map chain in connection with the re-baseline procedure shown in Figures 3A-3C.

[0025] DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0026] In the present invention, data is backed up continuously, allowing system administrators to pause, rewind, and replay live enterprise data streams. This moves the traditional backup methodologies into a continuous background process in which policies automatically manage the lifecycle of many generations of restore images.

[0027] System Construction

[0028] Figure 1A shows a preferred embodiment of a protected computer system 100 constructed in accordance with the present invention. A host computer 102 is connected directly to a primary data volume 104 (the primary data volume may also be referred to as the protected volume) and to a data protection system 106. The data protection system 106 manages a secondary data volume 108. The construction of the system 100 minimizes the lag time by writing directly to the primary data volume 104 and permits the data protection system 106 to focus exclusively on managing the secondary data volume 108. The management of the secondary volume 108 is preferably performed using a volume manager.

[0029] A volume manager is a software module that runs on a server or intelligent storage switch to manage storage resources. Typical volume managers have the ability to aggregate blocks from multiple different physical disks into one or more virtual volumes. Applications are not aware that they are actually writing to segments of many different disks because they are presented with one large, contiguous volume. In addition to block aggregation, volume managers usually also offer software RAID functionality. For example, they are able to split the segments of the different volumes into two groups, where one group is a mirror of the other group. This is, in a preferred embodiment, the feature that the data protection system is taking advantage of when the present invention is implemented as shown in Figure 1A. In many environments, the volume manager or host-based driver already mirrors the writes to two distinct different primary volumes for redundancy in case of a hardware failure. The present invention is configured as a tertiary mirror target in this scenario, such that the volume manager or host-based driver also sends copies of all writes to the data protection system.

[0030] It is noted that the primary data volume 104 and the secondary data volume 108 can be any type of data storage, including, but not limited to, a single disk, a disk array (such as a RAID), or a storage area network (SAN). The main difference between the primary data volume 104 and the secondary data volume 108 lies in the

structure of the data stored at each location, as will be explained in detail below. It is noted that there may also be differences in terms of the technologies that are used. The primary volume 104 is typically an expensive, fast, and highly available storage subsystem, whereas the secondary volume 108 is typically cost-effective, high capacity, and comparatively slow (for example, ATA/SATA disks). Normally, the slower secondary volume cannot be used as a synchronous mirror to the high-performance primary volume, because the slower response time will have an adverse impact on the overall system performance.

[0031] The data protection system 106, however, is optimized to keep up with high-performance primary volumes. These optimizations are described in more detail below, but at a high level, random writes to the primary volume 104 are processed sequentially on the secondary volume 108. Sequential writes improve both the cache behavior and the actual volume performance of the secondary volume 108. In addition, it is possible to aggregate multiple sequential writes on the secondary volume 108, whereas this is not possible with the random writes to the primary volume 104. The present invention does not require writes to the data protection system 106 to be synchronous. However, even in the case of an asynchronous mirror, minimizing latencies is important.

[0032] Figure 1B shows an alternate embodiment of a protected computer system 120 constructed in accordance with the present invention. The host computer 102 is directly connected to the data protection system 106, which manages both the primary data volume 104 and the secondary data volume 108. The system 120 is likely slower than the system 100 described above, because the data protection system 106 must manage both the primary data volume 104 and the secondary data volume 108. This results in a higher latency for writes to the primary volume 104 in the system 120 and lowers the available bandwidth for use. Additionally, the introduction of a new component into the primary data path is undesirable because of reliability concerns.

[0033] Figure 1C shows another alternate embodiment of a protected computer system 140 constructed in accordance with the present invention. The host computer

102 is connected to an intelligent switch 142. The switch 142 is connected to the primary data volume 104 and the data protection system 106, which in turn manages the secondary data volume 108. The switch 142 includes the ability to host applications and contains some of the functionality of the data protection system 106 in hardware, to assist in reducing system latency and improve bandwidth.

[0034] It is noted that the data protection system 106 operates in the same manner, regardless of the particular construction of the protected computer system 100, 120, 140. The major difference between these deployment options is the manner and place in which a copy of each write is obtained. To those skilled in the art it is evident that other embodiments, such as the cooperation between a switch platform and an external server, are also feasible.

[0035] Conceptual Overview

[0036] To facilitate further discussion, it is necessary to explain some fundamental concepts associated with a continuous data protection system constructed in accordance with the present invention. In practice, certain applications require continuous data protection with a block-by-block granularity, for example, to rewind individual transactions. However, the period in which such fine granularity is required is generally short (for example, two days), which is why the system can be configured to fade out data over time. The present invention discloses data structures and methods to manage this process automatically.

[0037] The present invention keeps a log of every write made to a primary volume (a "write log") by duplicating each write and directing the copy to a cost-effective secondary volume in a sequential fashion. The resulting write log on the secondary volume can then be played back one write at a time to recover the state of the primary volume at any previous point in time. Replaying the write log one write at a time is very time consuming, particularly if a large amount of write activity has occurred since the creation of the write log. In typical recovery scenarios, it is necessary to examine how the primary volume looked like at multiple points in time before

deciding which point to recover to. For example, consider a system that was infected by a virus. In order to recover from the virus, it is necessary to examine the primary volume as it was at different points in time to find the latest recovery point where the system was not yet infected by the virus. Additional data structures are needed to efficiently compare multiple potential recovery points.

[0038] Delta Maps

[0039] Delta maps provide a mechanism to efficiently recover the primary volume as it was at a particular point in time without the need to replay the write log in its entirety, one write at a time. In particular, delta maps are data structures that keep track of data changes between two points in time. These data structures can then be used to selectively play back portions of the write log such that the resulting point-in-time image is the same as if the log were played back one write at a time, starting at the beginning of the log.

[0040] Figure 2 shows a delta map 200 constructed in accordance with the present invention. While the format shown in Figure 2 is preferred, any format containing similar information may be used. For each write to a primary volume, a duplicate write is made, in sequential order, to a secondary volume. To create a mapping between the two volumes, it is preferable to have an originating entry and a terminating entry for each write. The originating entry includes information regarding the origination of a write, while the terminating entry includes information regarding the termination of a write.

[0041] As shown in delta map 200, row 210 is an originating entry and row 220 is a terminating entry. Row 210 includes a field 212 for specifying the region of a primary volume where the first block was written, a field 214 for specifying the block offset in the region of the primary volume where the write begins, a field 216 for specifying where on the secondary volume the duplicate write (i.e., the copy of the primary volume write) begins, and a field 218 for specifying the physical device (the physical volume or disk identification) used to initiate the write. Row 220 includes a field 222

for specifying the region of the primary volume where the last block was written, a field 224 for specifying the block offset in the region of the primary volume where the write ends, a field 226 for specifying the where on the secondary volume the duplicate write ends, and a field 228. While fields 226 and 228 are provided in a terminating entry such as row 220, it is noted that field 226 is optional because this value can be calculated by subtracting the offsets of the originating entry and the terminating entry (field 226 = (field 224 - field 214) + field 216), and field 228 is not necessary since there is no physical device usage associated with termination of a write.

[0042] In a preferred embodiment, as explained above, each delta map contains a list of all blocks that were changed during the particular time period to which the delta map corresponds. That is, each delta map specifies a block region on the primary volume, the offset on the primary volume, and physical device information. It is noted, however, that other fields or a completely different mapping format may be used while still achieving the same functionality. For example, instead of dividing the primary volume into block regions, a bitmap could be kept, representing every block on the primary volume. Once the retention policy (which is set purely according to operator preference) no longer requires the restore granularity to include a certain time period, corresponding blocks are freed up, with the exception of any blocks that may still be necessary to restore to later recovery points. Once a particular delta map expires, its block list is returned to the appropriate block allocator for re-use.

[0043] Delta maps are initially created from the write log using a map engine, and can be created in real-time, after a certain number of writes, or according to a time interval. It is noted that these are examples of ways to trigger the creation of a delta map, and that one skilled in the art could devise various other triggers. Additional delta maps may also be created as a result of a merge process (called "merged delta maps") and may be created to optimize the access and restore process. The delta maps are stored on the secondary volume and contain a mapping of the primary address space to the secondary address space. The mapping is kept in sorted order based on the primary address space.

[0044] One significant benefit of merging delta maps is a reduction in the number of delta map entries that are required. For example, when there are two writes that are adjacent to each other on the primary volume, the terminating entry for the first write can be eliminated from the merged delta map, since its location is the same as the originating entry for the second write. The delta maps and the structures created by merging maps reduces the amount of overhead required in maintaining the mapping between the primary and secondary volumes.

[0045] Dirty Region Log

[0046] The purpose of the dirty region log (DRL) mechanism is to track write operations that are in memory but not logged to the secondary volume. Depending on the particular deployment used (shown in Figure 1), the DRL is administered at a different location. For example, in the case of a host-based driver, the DRL can be located in the driver. Without some type of tracking mechanism, there would not be a record of these operations and a re-synchronization of the primary and secondary volumes after a failure would become an arduous process. The DRL does not keep track of where the primary volume data is being copied to on the secondary volume; it simply and efficiently tracks the locations on the primary volume that are being changed. The re-synchronization function can then use this information to narrow its scanning (i.e., compare and copy) operations to just those regions of the primary volume for which outstanding write operations not recorded in the write logs existed at the time of the failure.

[0047] The DRL is implemented in a preferred embodiment as a bit map with the bits representing equally sized regions of the primary volume address space. Each region typically contains multiple logical blocks of the primary volume. A bit set in the DRL indicates that a write operation has been (or at least may have been) initiated for the corresponding region of the primary volume. Since the completion status of the write operation is unknown, that region of the primary volume must be considered as potentially changed or "dirty."

[0048] For an example of the efficiency gained by using the DRL, assume that the primary volume is 100 GB and is divided into 1 GB blocks. Further assume that only one of the blocks is marked as dirty. If no DRL was used, the entire 100 GB of the primary volume would need to be compared with the secondary volume in order to resynchronize the two volumes. At a typical sequential access speed of 50 MB per second for a single disk, reading 100 GB of data on the primary volume would take over 33 minutes. In comparison, since the DRL indicates that only one block needs to be read, the read operation would only take 20 seconds. The need for such an optimization is more readily apparent when dealing with larger disk sizes typically encountered in an enterprise environment. As a second example, with 10 TB of primary storage, synchronizing the primary volume could take over two days (at a read speed of 50 MB per second) if no DRL is used.

[0049] During normal operation, write input/outputs (I/Os) are first tested against the current DRL to see if the region of the primary volume affected by the write operation is already "dirty." If it is, the write I/O is then immediately entered into the current write log and the write operations to the primary and secondary volumes are initiated. If the bit in the DRL corresponding to the write operation is clear, indicating that the region on the primary volume is "clean," then the bit is set (indicating a change to the primary data in that region) and the DRL is made persistent before the write operation is acknowledged. However, in an asynchronous implementation, the acknowledgement can occur before the secondary write is entered into the current write log. When the secondary volume acknowledges the write, the bit in the DRL is cleared.

[0050] The synchronous nature of the DRL change and commit is important. It is permissible for the DRL to have bits set even if data changes did not occur to the primary volume (e.g., the write I/O fails or is not issued due to a software failure), but it is not permissible for the DRL to have bits which are clear if there is a chance that the corresponding regions on the primary volume may have been changed.

[0051] Periodically as determined by policy, the DRL is reset by clearing all the bits in the map except for the bits representing regions of the primary volume for which uncommitted write I/Os exist (e.g., in write logs in memory but not yet written to disk). During certain critical phases of the re-synchronization workflow, it is important that the DRL be prevented from resetting. This is accomplished indirectly by manipulating the write log. The DRL reset policy will not allow the DRL to be reset when either the write log is not active or the write log is active but currently "Out of Sync." Deactivating the write log or inserting an "Out of Sync" marker into an active write log effectively disables DRL resets, while inserting an "In Sync" marker into an active write log effectively enables them.

[0052] During re-synchronization, regions on the primary volume corresponding to "dirty" bits in the DRL are compared to secondary volume data as mapped by the most recent reliable PIT map. The re-synchronization function can then bring the secondary volume data into agreement with the primary volume data if they are found to be different.

[0053] It is noted that there is a natural tension between the latency introduced by synchronous DRL writes (required when new regions of the primary volume are "dirtied") and the time required to re-synchronize the data after a failure. Over time, as the DRL fills with "dirty" bits, the fewer synchronous writes of the DRL are required, reducing the average I/O latency. However, the more filled the DRL becomes with "dirty" bits, the more regions of the primary volume must be scanned during re-synchronization. Resetting the DRL too frequently will reduce the time needed to re-synchronize at the expense of increased latency, while too few DRL resets will reduce latency at the expense of an overly long recovery time. The trade-off between latency and recovery time is handled by the DRL reset policy.

[0054] Re-Baseline

[0055] The re-baseline procedure is used to fill in the gaps where the data protection system, including the DRL, has failed. In general, the re-baseline procedure

operates similar to the creation of an initial snapshot, with the main difference being that only a fraction of the data has changed. Because certain optimizations exist, it is only necessary to fill in the gap in time where the failure occurred.

[0056] Over time, it is possible (although unlikely) that errors will accumulate within the data protection system with the result that the data on the primary and secondary volumes will no longer be in sync. The re-baseline procedure is a mechanism for detecting discrepancies between the primary and secondary volumes, marking suspect point-in-time intervals, and bringing the volumes back into synchronization. The re-baseline procedure can be run at any time to check the integrity of the primary and secondary volumes, and can fix any problems found in the background.

[0057] In general terms, the re-baseline procedure scans the entire primary and secondary volumes for data miscompares. When they are found, errors are corrected by copying data from the primary volume to the secondary volume and creating a "fix-up" delta map placed at the end of the delta map chain being maintained for the primary volume. The fix-up delta map brings the two volumes back into synchronization and restores any point in time (APIT) data protection. From this point in the map chain, the re-baseline procedure searches back through the map chain until it arrives at a map before which there are no errors. The interval (delta map sub-chain) between the earliest error-free map and the fix-up map is marked as being not reliable.

[0058] The re-baseline procedure introduces the concept of a "suspect" delta map, which is the system's way of delineating an interval of maps in the delta map chain through which errors have propagated. The suspect interval begins with the first map for which errors have been detected and continues forward in time until the fix-up delta map is reached. Each map within the interval is marked as suspect (except for the fix-up map), whether it contains mappings to regions with errors or not.

[0059] The significance of a suspect map is that if it is the most recent map in a map merge operation, the resulting merged map is also suspect. A suspect map merged with a more recent fix-up map results in a clean (non-suspect) map. A suspect map may only be merged with a more recent suspect map or a more recent fix-up map, i.e., it is

by definition impossible to have a more recent non-suspect map other than a fix-up map adjacent to a suspect map in the map chain. While merging two suspect maps would result in a suspect merged map, it is still desirable to create the most recent image of the delta map even in the suspect time period. The reason for this is that if a user attempts to recover the volume to a time within this period, the user will be presented with a warning, but will still be permitted to examine the volume. It is possible that the specific blocks or files the user is looking for have not been corrupted. The map manager uses the suspect map marker to keep track of how reliable various composite maps are when they are built from the fundamental delta map chain elements.

[0060] Figures 3A-3C show flowcharts describing the steps of the re-baseline procedure 300, which is also graphically represented from a delta map perspective in Figure 4. The following discussion uses all of these figures to explain the re-baseline procedure 300.

[0061] The re-baseline procedure 300 begins (step 302; point B in Figure 4) by creating a current delta map covering from the last delta map in the map chain to the present time (step 304). A current PIT map is created by merging all of the delta maps in the map chain prior to and including the current delta map (step 306); this process may be optimized by utilizing pre-merged delta maps. The PIT map is used for comparing the primary volume data and secondary volume data as they exist at the present time. While the PIT map is utilized, the entire primary volume still needs to be scanned to properly complete the re-baseline procedure.

[0062] A miscompare delta map (Δmc) is created to track any miscompares (step 308). A region of the primary volume is scanned (step 310) and the scanned region is compared with the corresponding region of the secondary volume as indicated by the PIT map (step 312). This part of the process (steps 310 and 312) can be optimized in some situations by utilizing the DRL. Care must be taken when dealing with suspect volumes, because the DRL may not provide accurate information, and therefore cannot be used. If the DRL is current, then the re-synchronization procedure can be used, as

discussed below. The safest method is to scan the entire volume, especially since the scanning can be performed in the background while the system is processing other requests. It is possible to use checkpoints such that one region of the volumes can be scanned and compared at a time, and if the system fails for some reason during the scan and compare steps, it will not be necessary to restart the entire re-baseline procedure from the beginning.

[0063] A determination is made whether the regions on the primary and secondary volumes match (step 314). If the regions do not match, this indicates that there is a miscompare between the primary and secondary volumes, and an entry is made into the miscompare delta map (step 316). If the regions match (step 314), then there are no miscompares between the two regions. Regardless of the outcome of step 314, another decision is made whether all of the regions of the primary and secondary volumes have been compared (step 318). As noted above, the DRL may be used to optimize this decision by only looking at the regions marked as "dirty" in the DRL. If all of the regions on the primary volume have not been compared, then the procedure continues at step 310.

[0064] If all of the regions on the primary volume have been compared (step 318), then a scan delta map (Δscan) is created for the scan interval (between points B and C in Figure 4; step 320). The scan delta map is created by merging any delta maps that exist from the time the re-baseline procedure was started and the present time. Because host write activity has continued during the scan interval, some of the changes to the primary volume data may have resulted in miscompares and corresponding entries in the miscompare delta map. These miscompares do not represent errors (they represent new writes to the primary volume) and need to be removed from the miscompare delta map.

[0065] A copy delta map (Δcopy) is created to store only the true miscompares between the primary and secondary volumes (step 322). In order to bring the primary and secondary volumes into agreement, data must be copied from the primary volume to the secondary volume for those areas where true miscompares have been identified.

The copy delta map is used to guide the copy process by factoring out any changes made to the primary volume during the scan interval that may have resulted in false miscompares. This map is constructed by subtracting from the miscompare map (Δmc) the intersection of the miscompare map (Δmc) and the scan interval map (Δscan), as shown in Equation 1.

$$\Delta copy = \Delta mc - (\Delta mc \cap \Delta scan) \qquad \text{Equation (1)}$$

[0066] A determination is made whether the copy delta map is empty (step 324). If the copy delta map is empty, indicating that there are no true miscompares between the primary and secondary volumes (i.e., that the primary and secondary volumes are synchronized), then the procedure terminates (step 326).

[0067] If the copy delta map is not empty (step 324), then data from the primary volume is copied onto the secondary volume for those areas where miscompares were detected, overwriting the areas on the secondary volume, using the copy delta map and the current PIT map (step 328; between points C and D in Figure 4). A host change delta map (Δhost) is created for the copy interval to track the new changes made during the copy interval (step 330).

[0068] A fix-up delta map (Δfix-up) is created by merging the copy delta map into the host change delta map (step 332), as shown in Equation 2.

$$\Delta fix\text{-}up = \Delta host \leftarrow \Delta copy \qquad \text{Equation (2)}$$

[0069] The fix-up delta map is then added to the end of the delta map chain (step 334). The fix-up delta map is used to bring the primary and secondary volumes into synchronization since it includes all of the changes made to the primary volume from the beginning of the scan interval (point B in Figure 4) and to restore the map chain to reliable PIT capability (point D in Figure 4).

[0070] Because the copy delta map is not empty, errors existed in the delta map chain prior to the beginning of the scan interval (point B in Figure 4), and all delta maps within the scan interval are suspect. The first delta map before the start of the

scan interval (to the left of point B in Figure 4) is selected (step 336). The selected delta map is marked as suspect (step 338). The copy delta map is adjusted by subtracting the changes in the selected delta map (Δselected) from the copy delta map (step 340) and as shown in Equation 3.

$$\Delta copy = \Delta copy - (\Delta copy \cap \Delta selected) \qquad \text{Equation (3)}$$

[0071] A determination is made whether the adjusted copy delta map is now empty (step 342). If the copy delta map is empty, indicating that there are no additional suspect delta maps to locate, then the procedure terminates (step 326). If the copy delta map is not empty (step 342), then the previous delta map in the map chain is selected (step 344) and the procedure continues with step 338. The point in the delta map chain preceding the earliest delta map to be marked as suspect (point A in Figure 4) is the beginning of the suspect interval for the map chain.

[0072] Re-Synchronization

[0073] If the DRL is correct (meaning that the data protection system has not failed) and the secondary volume fails, then a re-synchronization procedure can be used to update the secondary volume. The re-synchronization procedure operates in the same manner as the re-baseline procedure described above in connection with Figures 3A-3C, with the main difference being that the re-synchronization procedure is able to use the DRL to optimize the scan and compare steps (steps 310 and 312). As a result of these optimizations, the scan interval will be shorter since only those regions contained in the DRL need to be scanned and compared.

[0074] Full Restore

[0075] The full restore procedure is used to place a snapshot onto the primary volume, in the event of a primary volume failure. Data is copied from the snapshot into the primary volume. It is possible to roll forward changes in the data from the time of the snapshot to the present time, at the user's preference. With appropriate file system

decoders, it is even possible to selectively apply changes to certain data (i.e., a user can roll forward changes on a particular file), so the full restore procedure may function like a concurrent version system, as is used in software development.

[0076] The idea of a full restore is that once a user has selected a particular snapshot, the corresponding content (i.e., the new volume) is moved back to the primary volume, overwriting the corrupted contents there. If a host-based agent or service on a storage switch is used, requests to regions that have already been copied back to the primary volume can be served from there (and thus faster), whereas requests for blocks that have not been copied are served from the secondary volume. This makes the restore process invisible to the user, with the only noticeable difference being that initial requests will be a bit slower than usual.

[0077] While specific embodiments of the present invention have been shown and described, many modifications and variations could be made by one skilled in the art without departing from the scope of the invention. The above description serves to illustrate and not limit the particular invention in any way.

* * *